LocNet: Improving Localization Accuracy for Object Detection

Spyros Gidaris, Nikos Komodakis

LocNet: Improving Localization Accuracy for Object Detection

Spyros Gidaris
Universite Paris Est, Ecole des Ponts ParisTech
gidariss@imagine.enpc.fr

Nikos Komodakis
Universite Paris Est, Ecole des Ponts ParisTech
nikos.komodakis@enpc.fr

Abstract

We propose a novel object localization methodology with the purpose of boosting the localization accuracy of state-of-the-art object detection systems. Our model, given a search region, aims at returning the bounding box of an object of interest inside this region. To accomplish its goal, it relies on assigning conditional probabilities to each row and column of this region, where these probabilities provide useful information regarding the location of the boundaries of the object inside the search region and allow the accurate inference of the object bounding box under a simple probabilistic framework.

For implementing our localization model, we make use of a convolutional neural network architecture that is properly adapted for this task, called LocNet. We show experimentally that LocNet achieves a very significant improvement on the mAP for high IoU thresholds on PASCAL VOC2007 test set and that it can be very easily coupled with recent state-of-the-art object detection systems, helping them to boost their performance. Furthermore, it sets a new state-of-the-art on PASCAL VOC2012 test set achieving mAP of 74.8%*. Finally, we demonstrate that our detection approach can achieve high detection accuracy even when it is given as input a set of sliding windows, thus proving that it is independent of bounding box proposal methods.*

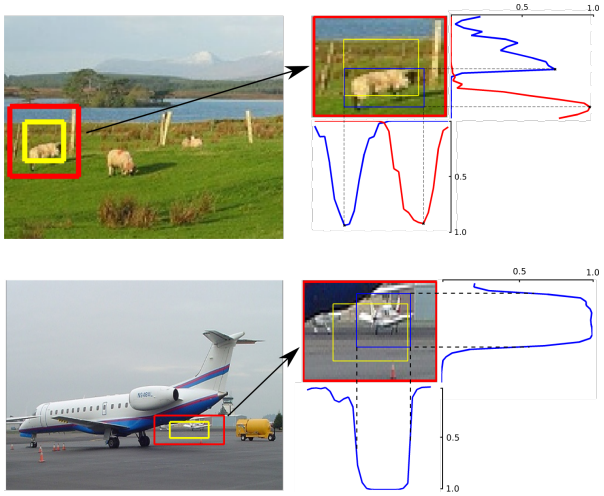


Figure 1: Illustration of the basic work-flow of our localization module. ***Left column:*** our model given a candidate box B (yellow box) it "looks" on a search region R (red box), which is obtained by enlarging box B by a constant factor, in order to localize the bounding box of an object of interest. ***Right column:*** To localize a bounding box the model assigns one or more probabilities on each row and independently on each column of region R. Those probabilities can be either the probability of an element (row or column) to be one of the four object borders (see top-right image), or the probability for being on the inside of an objects bounding box (see bottom-right image). In either case the predicted bounding box is drawn with blue color.

1. Introduction

Object detection is one of the most challenging and well studied problems in computer vision that has attracted an immense amount of attention especially over the last years. One of the main challenges in this case is how to improve the localization accuracy by which a detection system is able to predict the bounding boxes of the objects of interest, where accuracy is measured by the Intersection over Union (IoU) between the predicted and the ground truth bounding box. Although an IoU detection theshold of 0.5 is used in challenges such as PASCAL VOC, a higher localization accuracy (e.g. IoU $\geqslant$ 0.7) is often required for many real life applications, such as for robotic arms that need to grab objects. Such a need is also reflected in the very recently introduced *COCO* detection challenge [21], which uses as evaluation metric the traditional average precision (AP) measurement but averaged over multiple IoU thresholds between 0.5 (loosely localized object) and 1.0 (perfectly localized object) so as to reward detectors that exhibit good localization accuracy.

It thus seems that one significant future challenge on ob-

This work was supported by the ANR SEMAPOLIS project. We are planning to release our code and trained models in order to support progress in the area of object detection and recognition and to allow others to experiment with it.

ject detection will be on proposing detectors that exhibit accurate localization of the ground truth objects. This important aspect is exactly the focus of this work, which, to the best of our knowledge, is one of the first that targets this direction. In practical terms, our goal is to boost the bounding box detection AP performance across a wide range of IoU thresholds (*i.e.*, not just for IoU threshold of 0.5 but also for values well above that). To that end, a main contribution of this work is to propose a novel *object localization model* that, given a loosely localized search region inside an image, aims to return the accurate location of an object in this region (see Figure 1). Importantly, such a localization module can be easily incorporated into many of the current state-of-the-art object detection systems [7, 8, 26], helping them to significantly improve their localization performance. Here we use it in an iterative manner as part of a detection pipeline that utilizes a recognition model for scoring candidate bounding boxes provided by the aforementioned localization module, and show that such an approach significantly boosts AP performance across a broad range of IoU thresholds.

Related work. Most of the recent literature on object detection, treats the object localization problem at pre-recognition level by incorporating category-agnostic object proposal algorithms [33, 37, 24, 1, 17, 16, 2, 32, 31] that given an image, try to generate candidate boxes with high recall of the ground truth objects that they cover. Those proposals are later classified from a category-specific recognition model in order to create the final list of detections [9]. Instead, in our work we focus on boosting the localization accuracy at post-recognition time, at which the improvements can be complementary to those obtained by improving the pre-recognition localization. Till now, the work on this level has been limited to the bounding box regression paradigm that was first introduced from Felzenszwalb *et al*. [6] and ever-since it has been used with success on most of the recent detection systems [9, 8, 26, 28, 11, 34, 36, 27, 22]. A regression model, given an initial candidate box that is loosely localized around an object, it tries to predict the coordinates of its ground truth bounding box. Lately this model is enhanced by high capacity convolutional neural networks to further improve its localization capability [7, 8, 28, 26].

Contributions. The motivation behind our work stems from the belief that trying to directly regress to the target bounding box coordinates, constitutes a difficult learning task that cannot yield accurate enough bounding boxes. We argue that it is far more effective to attempt to localize a bounding box by first assigning a probability to each row and independently to each column of the search region for being the left, right, top, or bottom borders of the bounding box (see Figure 1) or for being on the inside of an objects bounding box (see Figure 1). Those probabilities can provide a measure of confidence for placing the bounding box on each location and can also handle instances that exhibit multi-modal distributions for the border locations. They thus yield far more detailed and useful information than the regression models that just predict 4 real values that correspond to estimations of the bounding box coordinates. As a result of this, we argue also that the task of learning to predict these probabilities is an easier one to accomplish. To implement the proposed localization model, we rely on a convolutional neural network model, which we call *LocNet*, whose architecture is properly adapted such that the amount of parameters needed on the top fully connected layers is significantly reduced, thus making our LocNet model scalable with respect to the number of object categories.

To summarize, our contributions are as follows:

- We cast the problem of localizing an object's bounding box as that of assigning probabilities on each row and column of a search region. Those probabilities represent either the likelihood of each element (row or column) to belong on the inside of the bounding box or the likelihood to be one of the four borders of the object. Both of those cases is studied and compared with the bounding box regression model.
- To implement the above model, we propose a properly adapted convolutional neural network architecture that has a reduced number of parameters and results in an efficient and accurate object localization network (LocNet).
- We extensively evaluate our approach on VOC2007 [4] and we show that it achieves a very significant improvement over the bounding box regression with respect to the mAP for IoU threshold of 0.7 and the COCO style of measuring the mAP. It also offers an improvement with respect to the traditional way of measuring the mAP (*i.e.*, for IoU $\geqslant 0.5$), achieving in this case 78.4% and 74.78% mAP on VOC2007 [4] and VOC2012 [5] detection challenge, which is the state-of-the-art at the time of writing this paper. Given those results we believe that our localization approach could very well replace the existing bounding box regression paradigm in future object detection systems.
- Finally we demonstrate that the detection accuracy of our system remains high even when it is given as input a set of *sliding windows*, which proves that it is independent of bounding box proposal methods if the extra computational cost is ignored.

The remainder of the paper is structured as follows: We describe our object detection methodology in §2 and then present our localization model in §3. Implementation details and experimental results are provided in §4 and §5 respectively. Finally, we conclude in §6.

2. Object Detection Methodology

Algorithm 1: Object detection pipeline

Input : Image $\mathbf{I}$, initial set of candidate boxes $\mathbf{B}^1$
Output: Final list of detections $\mathbf{Y}$
for $t \leftarrow 1$ ***to*** T **do**
  $\mathbf{S}^t \leftarrow \mathit{Recognition}(\mathbf{B}^t|\mathbf{I})$
  if $t < T$ **then**
    $\mathbf{B}^{t+1} \leftarrow \mathit{Localization}(\mathbf{B}^t|\mathbf{I})$
  end
end
$\mathbf{D} \leftarrow \cup_{t=1}^{T}\{\mathbf{S}^t, \mathbf{B}^t\}$
$\mathbf{Y} \leftarrow \mathit{PostProcess}(\mathbf{D})$

Our detection pipeline includes two basic components, the recognition and the localization models, integrated into an iterative scheme (see algorithm 1). This scheme starts from an initial set of candidate boxes $\mathbf{B}^1$ (which could be, *e.g.*, either dense sliding windows [28, 23, 25, 20] or category-agnostic bounding box proposals [37, 33, 26]) and on each iteration t it uses the two basic components in the following way:

Recognition model: Given the current set of candidate boxes $\mathbf{B}^t = \{B_i^t\}_{i=1}^{N_t}$, it assigns a confidence score to each of them $\{s_i^t\}_{i=1}^{N_t}$ that represents how likely it is for those boxes to be localized on an object of interest.

Localization model: Given the current set of candidate boxes $\mathbf{B}^t = \{B_i^t\}_{i=1}^{N_t}$, it generates a new set of candidate boxes $\mathbf{B}^{t+1} = \{B_i^{t+1}\}_{i=1}^{N_{t+1}}$ such that those boxes they will be "closer" (*i.e.*, better localized) on the objects of interest (so that they are probably scored higher from the recognition model).

In the end, the candidate boxes that were generated on each iteration from the localization model along with the confidences scores that were assigned to them from the recognition model are merged together and a post-processing step of non-max-suppression [6] followed from bounding box voting [7] is applied to them. The output of this post-processing step consists the detections set produced from our pipeline. Both the recognition and the localization models are implemented as convolutional neural networks [19] that lately have been empirically proven quite successful on computers vision tasks and especially those related to object recognition problems [29, 18, 12, 14, 30]. More details about our detection pipeline are provided in appendix D.

3. Localization model

In this paper we focus on improving the localization model of this pipeline. The abstract work-flow that we use for this model is that it gets as input a candidate box B in the image, it enlarges it by a factor γ to create a search region R and then it returns a new candidate box that ideally will tightly enclose an object of interest in this region (see right column of Figure 1).

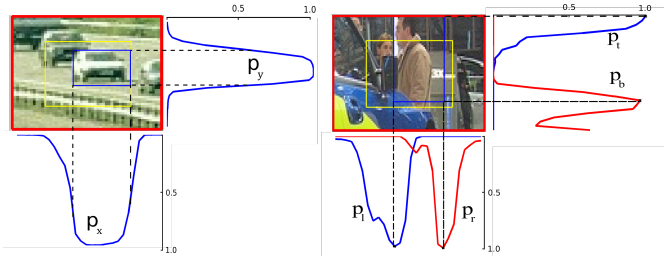


Figure 2: The posterior probabilities that our localization model yields given a region R. ***Left Image:*** the in-out conditional practicabilities that are assigned on each row (p_y) and column (p_x) of R. They are drawn with the blues curves on the right and on the bottom side of the search region. ***Right Image:*** the conditional probabilities p_l, p_r, p_t, and p_b of each column or row to be the left (l), right (r), top (t) and bottom (b) border of an object's bounding box. They are drawn with blue and red curves on the bottom and right side of the search region.

The crucial question is, of course, what is the most effective approach for constructing a model that is able to generate a good box prediction. One choice could be, for instance, to learn a regression function that directly predicts the 4 bounding box coordinates. However, we argue that this is not the most effective solution. Instead, we opt for a different approach, which is detailed in the next section.

3.1. Model predictions

Given a search region R and object category c, our object localization model considers a division of R in M equal horizontal regions (rows) as well as a division of R in M equal vertical regions (columns), and outputs for each of them one or more conditional probabilities. Each of these conditional probabilities is essentially a vector of the form $p^{R,c} = \{p(i|R,c)\}_{i=1}^{M}$ (hereafter we drop the R and c conditioned variables so as to reduce notational clutter). Two types of conditional probabilities are considered in this work:

In-Out probabilities: These are vectors $p_x = \{p_x(i)\}_{i=1}^{M}$ and $p_y = \{p_y(i)\}_{i=1}^{M}$ that represent respectively the conditional probabilities of each column and row of R to be inside the bounding box of an object of category c (see left part of Figure 2). A row or column is considered to be inside a bounding box if at least part of the region corresponding to this row or column is inside this box. For example, if B^{gt} is a ground truth bounding box with top-left coordi-

We use $\gamma = 1.8$ in all of the experiments.

nates (B_l^{gt}, B_t^{gt}) and bottom-right coordinates (B_r^{gt}, B_b^{gt}), then the In-Out probabilities $p = \{p_x, p_y\}$ from the localization model should ideally equal to the following target probabilities $T = \{T_x, T_y\}$:

$$\forall i \in \{1, \dots, M\}, \;\; T_x(i) = \begin{cases} 1, & \text{if } B_l^{gt} \leq i \leq B_r^{gt} \\ 0, & \text{otherwise} \end{cases},$$

$$\forall i \in \{1, \dots, M\}, \;\; T_y(i) = \begin{cases} 1, & \text{if } B_t^{gt} \leq i \leq B_b^{gt} \\ 0, & \text{otherwise} \end{cases}.$$

Border probabilities: These are vectors $p_l = \{p_l(i)\}_{i=1}^M$, $p_r = \{p_r(i)\}_{i=1}^M$, $p_t = \{p_t(i)\}_{i=1}^M$ and $p_b = \{p_b(i)\}_{i=1}^M$ that represent respectively the conditional probability of each column or row to be the left (l), right (r), top (t) and bottom (b) border of the bounding box of an object of category c (see right part of Figure 2). In this case, the target probabilities $T = \{T_l, T_r, T_t, T_b\}$ that should ideally be predicted by the localization model for a ground truth bounding box $B^{gt} = (B_l^{gt}, B_t^{gt}, B_r^{gt}, B_b^{gt})$ are given by

$$\forall i \in \{1, \dots, M\}, \;\; T_s(i) = \begin{cases} 1, & \text{if } i = B_s^{gt} \\ 0, & \text{otherwise} \end{cases},$$

where $s \in \{l, r, t, b\}$. Note that we assume that the left and right border probabilities are independent and similarly for the top and bottom cases.

3.1.1 Bounding box inference

Given the above output conditional probabilities, we model the inference of the bounding box location $B = (B_l, B_t, B_r, B_b)$ using one of the following probabilistic models:

In-Out ML: Maximizes the likelihood of the *in-out* elements of B

$$L_{\text{in-out}}(B) = \prod_{i \in \{B_l, \dots, B_r\}} p_x(i) \prod_{i \in \{B_t, \dots, B_b\}} p_y(i) \prod_{i \notin \{B_l, \dots, B_r\}} \tilde{p}_x(i) \prod_{i \notin \{B_t, \dots, B_b\}} \tilde{p}_y(i), \quad (1)$$

where $\tilde{p}_x(i) = 1 - p_x(i)$ and $\tilde{p}_y(i) = 1 - p_y(i)$. The first two terms in the right hand of the equation represent the likelihood of the rows and columns of box B (*in*-elements) to be inside a ground truth bounding box and the last two terms the likelihood of the rows and columns that are not part of B (*out*-elements) to be outside a ground truth bounding box.

Borders ML: Maximizes the likelihood of the borders of box B:

$$L_{\text{borders}}(B) = p_l(B_l) \cdot p_t(B_t) \cdot p_r(B_r) \cdot p_b(B_b). \quad (2)$$

We actually assume that the ground truth bounding box is projected on the output domain of our model where the coordinates take integer values in the range $\{1, \dots, M\}$. This is a necessary step for the definition of the target probabilities

Combined ML: It uses both types of probability distributions by maximizing the likelihood for both the *borders* and the *in-out* elements of B:

$$L_{\text{combined}}(B) = L_{\text{borders}}(B) \cdot L_{\text{in-out}}(B). \quad (3)$$

3.1.2 Discussion

The reason we consider that the proposed formulation of the problem of localizing an object's bounding box is superior is because the In-Out or Border probabilities provide much more detailed and useful information regarding the location of a bounding box compared to the typical bounding box regression paradigm [6]. In particular, in the later case the model simply directly predicts real values that corresponds to estimated bounding box coordinates but it does not provide, *e.g.*, any confidence measure for these predictions. On the contrary, our model provides a conditional probability for placing the four borders or the inside of an object's bounding box on each column and row of a search region R. As a result, it is perfectly capable of handling also instances that exhibit multi-modal conditional distributions (both during training and testing). During training, we argue that this makes the per row and per column probabilities much easier to be learned from a convolutional neural network that implements the model, than the bounding box regression task (*e.g.*, see Figure 3), thus helping the model to converge to a better training solution. Indeed, as we demonstrate, *e.g.*, in Figure 4, our CNN-based *In-Out ML* localization model converges faster and on higher localization accuracy (measured with the mAR [13] metric) than a CNN-based bounding box regression model [8, 7]. This behaviour was consistently observed in all of our proposed localization models.

Furthermore, during testing, these conditional distributions as we saw can be exploited in order to form probabilistic models for the inference of the bounding box coordinates. In addition, they can indicate the presence of a second instance inside the region R and thus facilitate the localization of multiple adjacent instances, which is a difficult problem on object detection. In fact, when visualizing, *e.g.*, the border probabilities, we observed that this could have been possible in several cases (*e.g.*, see Figure 5). Although in this work we did not explore the possibility of utilizing a more advanced probabilistic model that predicts $K > 1$ boxes per region R, this can certainly be an interesting future addition to our method.

Alternatively to our approach, we could predict the probability of each pixel to belong on the foreground of an object, as Pinheiro *et al.* [24] does. However, in order to learn such a type of model, pixel-wise instance segmentation masks are required during training, which in general is a rather tedious task to collect. In contrary, for our model to learn those per row and per column probabilities, only

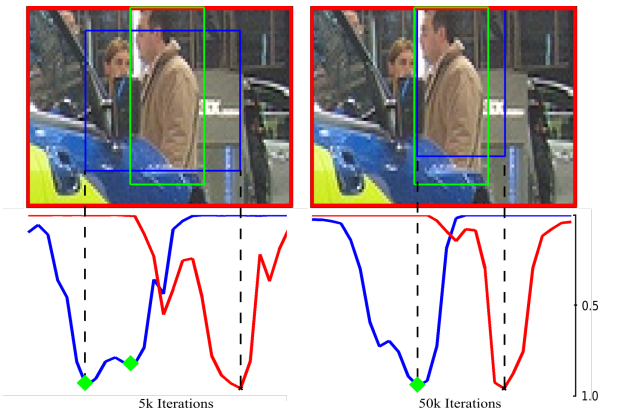


Figure 3: We show the evolution during training. In the left image the green squares indicate the two highest modes of the left border probabilities predicted by a network trained only for a few iterations (5k). Despite the fact that the highest one is erroneous, the network also maintains information for the correct mode. As training progresses (50k), this helps the network to correct its mistake and recover a correct left border(right image).

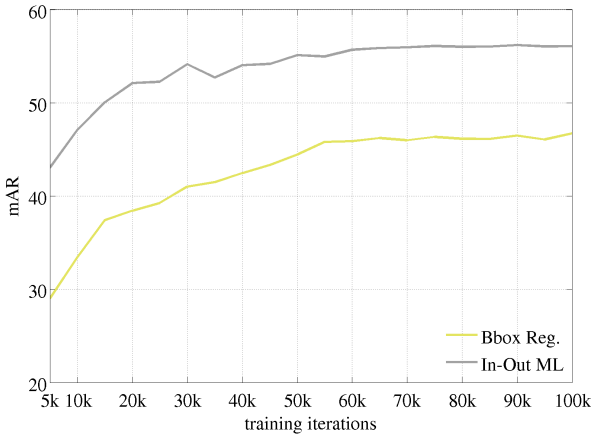


Figure 4: mAR as a function of the training iteration for the bounding box regression model (*Bbox reg.*) and the *In-Out ML* localization model. In order to create this plot, we created a small validation set of candidate boxes with a ground truth bounding box assigned on each of them, and during training given those candidates as input to the models we measure the mAR of the predicted boxes. We observe that the *In-Out ML* localization model converges faster and to a higher mAR than the *bounding box regression* localization model.

bounding box annotations are required. Even more, this independence is exploited in the design of the convolutional neural network that implements our model in order to keep the number of parameters of the prediction layers small (see § 3.2). This is significant for the scalability of our model with respect to the number of object categories since we favour category-specific object localization that has been shown to exhibit better localization accuracy [29].

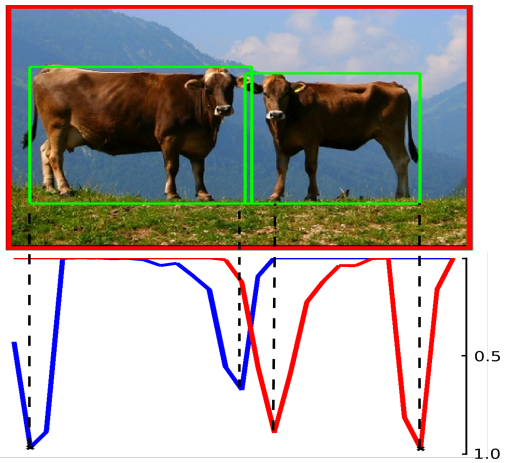


Figure 5: We depict the probabilities for the left (blue) and right (red) borders that a trained model yields for a region with two instances of the same class (cow). The probability modes in this case can clearly indicate the presence of two instances.

3.2. LocNet network architecture

Our localization model is implemented through the convolutional neural network that is visualized in Figure 6 and which is called LocNet. The processing starts by forwarding the entire image I (of size $w_I \times h_I$), through a sequence of convolutional layers (conv. layers of VGG16 [29]) that outputs the A_I activation maps (of size $\frac{w_I}{16} \times \frac{h_I}{16} \times 512$). Then, the region R is projected on A_I and the activations that lay inside it are cropped and pooled with a spatially adaptive max-pooling layer [11]. The resulting fixed size activation maps ($14 \times 14 \times 512$) are forwarded through two convolutional layers (of kernel size $3 \times 3 \times 512$), each followed by ReLU non-linearities, that yield the localization-aware activation maps A_R of region R (with dimensions size $14 \times 14 \times 512$).

At this point, given the activations A_R the network yields the probabilities that were described in section §3.1. Specifically, the network is split into two branches, the *X* and *Y*, with each being dedicated for the predictions that correspond to the dimension (x or y respectively) that is assigned to it. Both start with a max-pool layer that aggregates the A_R activation maps across the dimension perpendicular to the one dedicated to them, *i.e.*,

$$A_R^x(i,f) = \max_j A_R(i,j,f), \tag{4}$$

$$A_R^y(j,f) = \max_i A_R(i,j,f), \tag{5}$$

where i,j,and f are the indices that span over the width, height, and feature channels of A_R respectively. The resulted activations A_R^x and A_R^y (both of size 14×512) efficiently encode the object location only across the dimension that their branch handles. This aggregation process could

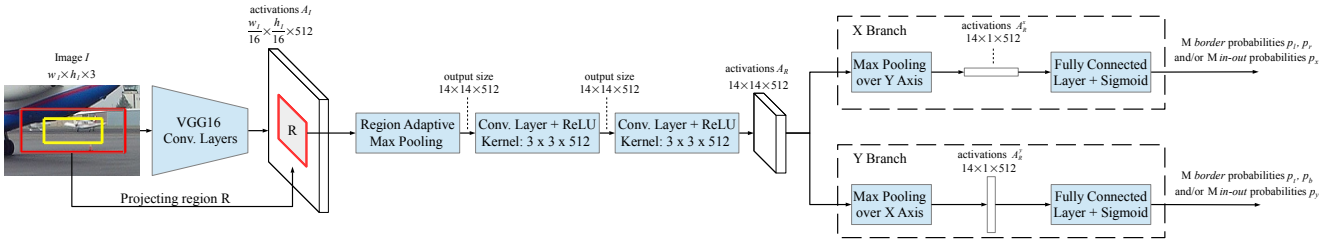


Figure 6: Visualization of the LocNet network architecture. In the input image, with yellow is drawn the candidate box B and with red the search region R. In its output, the LocNet network yields probabilities for each of the C object categories. The parameter M that controls the output resolution is set to the value 28 in our experiments. The convolutional layers of the VGG16-Net [29] that are being used in order to extract the image activations A_I are those from conv1_1 till conv5_3. The new layers that are not derived from the VGG16-Net [29], are randomly initialized with a Gaussian distribution with standard deviation of 0.001 for the hidden layers and 0.01 for the final fully connected layers.

also be described as marginalizing-out localization cues irrelevant for the dimension of interest. Finally, each of those aggregated features is fed into the final fully connected layer that is followed from sigmoid units in order to output the conditional probabilities of its assigned dimension. Specifically, the *X* branch outputs the p_x and/or the (p_l, p_r) probability vectors whereas the *Y* branch outputs the p_y and/or the (p_t, p_b) probability vectors. Despite the fact that the last fully connected layers output category-specific predictions, their number of parameters remains relatively small due to the facts that: 1) they are applied on features of which the dimensionality has been previously dramatically reduced due to the max-pooling layers of equations 4 and 5, and 2) that each branch yields predictions only for a single dimension.

3.3. Training

During training, the network learns to map a search regions R to the target probabilities T that are conditioned on the object category c. Given a set of N training samples $\{(R_k, T_k, c_k)\}_{k=1}^{N}$ the loss function that is minimised is

$$L(\theta) = \frac{1}{N} \sum_{k=1}^{N} l(\theta | R_k, T_k, c_k), \tag{6}$$

where θ are the network parameters that are learned and $l(\theta|R, T, c)$ is the loss for one training sample.

Both for the *In-Out* and the *Borders* probabilities we use the sum of binary logistic regression losses per row and column. Specifically, the per sample loss of the *In-Out* case is:

$$\sum_{a \in \{x,y\}} \sum_{i=1}^{M} T_a(i) \log(p_a(i)) + \tilde{T}_a(i) \log(\tilde{p}_a(i)), \tag{7}$$

and for the *Borders* case is:

$$\sum_{s \in \{l,r,u,b\}} \sum_{i=1}^{M} \lambda^+ T_s(i) \log(p_s(i)) + \lambda^- \tilde{T}_s(i) \log(\tilde{p}_s(i)), \tag{8}$$

where $\tilde{p} = 1 - p$. In objective function (8), λ^+ and λ^- represent the weightings of the losses for misclassifying a border and a non-border element respectively. These are set as

$$\lambda^- = 0.5 \cdot \frac{M}{M-1}, \quad \lambda^+ = (M-1) \cdot \lambda^-,$$

so as to balance the contribution on the loss of those two cases (note that $\tilde{T}_s(i)$ will be non-zero $M-1$ times more than $T_s(i)$). We observed that this leads to a model that yields more "confident" probabilities for the borders elements. For the *Borders* case we also tried to use as loss function the Mean Square Error, while modifying the target probabilities to be Gaussian distributions around the border elements, but we did not observe an improvement in performance.

4. Implementation details

General: For the implementation code of our paper we make use of the CAFFE framework [15]. During training of all the models (both the localization and the recognition ones) we fine-tune only from the conv4_1 convolutional layer and above. As training samples we use both selective search [33] and edge box [37] proposals. Finally, both during training and testing we use a single image scale that is obtained after resizing the image such as its smallest dimension to be 600 pixels.

Proposed localization models (*In-Out ML*, *Borders ML*, *Combined ML*): To create the training samples we take proposals of which the IoU with a ground truth bounding box is at least 0.4, we enlarge them by a factor of 1.8 in order to obtain the search regions R, and we assign to them the ground truth bounding box with which the original box proposal has the highest IoU in order to obtain the target bounding boxes and the corresponding target vectors T. This process is performed independently for each category. The parameter M that controls the output resolution

Initial set of candidate boxes	Number	Recall		
		IoU≥0.5	IoU≥0.7	mAR
Sliding Windows	around 10k	0.920	0.389	0.350
Edge Box	around 2k	0.928	0.755	0.517
Sel. Search	around 2k	0.936	0.687	0.528

Table 1: Recall statistics of the box proposals methods that we use in our work in order to generate the initial set of candidate boxes. The reported results are for the VOC2007 test set.

of our networks, is set to the value 28. For optimization we use stochastic gradient descend (SGD) with mini-batch size of 128 training candidate boxes. To speed up the training procedure we follow the paradigm of Fast-RCNN [8] and those 128 training candidate boxes are coming from only two images on each mini-batch. The weight decay is set to 0.00005 and the learning rate is set to 0.001 and is reduced by a factor of 10 after each $60k$ iterations. The overall training procedure is continued for up to $150k$ iterations and it takes around 1.5 days in one NVIDIA Titan Black GPU.

5. Experimental results

We empirically evaluate our localization models on PASCAL VOC detection challenge. Specifically, we train all the recognition and localization models on VOC2007+2012 trainval sets and we test them on the VOC2007 test set. As baseline we use a CNN-based bounding box regression model [7] (see appendices A and B). The remaining components of the detection pipeline include

Initial set of candidate boxes: We examine three alternatives for generating the initial set of candidate boxes: the Edge Box algorithm [37] (*EB*), the Selective Search algorithm (*SS*), and a sliding windows scheme. In Table 1 we provide the recall statistics of the initial bounding box proposal methods.

Recognition model: For the recognition part of the detection system we use either the *Fast-RCNN [8]* or the *MR-CNN [7]* recognition models. During implementing the latter one, we performed several simplifications on its architecture and thus we call the resulting model *Reduced-MR-CNN* (those modifications are detailed in appendix C). The Fast-RCNN and Reduced-MR-CNN models are trained using both selective search and edge box proposals and as top layer they have class-specific linear SVMs [9].

First, we examine the performance of our approach with respect to localization (§5.1) and detection (§5.2) accuracy. Then we report the detection accuracy of our approach for the sliding windows case (§5.3). Finally, we provide qualitative results in §5.4.

5.1. Localization performance

We first evaluate merely the localization performance of our models, thus ignoring in this case the recognition aspect of the detection problem. For that purpose we report the recall that the examined models achieve. Specifically, in Figure 7 we provide the recall as a function of the IoU threshold for the candidate boxes generated on the first iteration and the last iteration of our detection pipeline. Also, in the legends of these figures we report the average recall (AR) [13] that each model achieves. Note that, given the set of initial candidate boxes and the recognition model, the input to the iterative localization mechanism is exactly the same and thus any difference on the recall is solely due to the localization capabilities of the models. We observe that for IoU thresholds above 0.65, the proposed models achieve higher recall than bounding box regression and that this improvement is actually increased with more iterations of the localization module. Also, the AR of our proposes models is on average 6 points higher than bounding box regression.

5.2. Detection performance

Here we evaluate the detection performance of the examined localization models when plugged into the detection pipeline that was described in section §2. In Table 2 we report the mAP on VOC2007 test set for IoU thresholds of 0.5 and 0.7 as well as the COCO style of mAP that averages the traditional mAP over various IoU thresholds between 0.5 and 1.0. The results that are reported are obtained after running the detection pipeline for $T = 4$ iterations. We observe that the proposed *InOut ML*, *Borders ML*, and *Combined ML* localization models offer a significant boost on the mAP for IoU ≥ 0.7 and the COCO style mAP, relative to the bounding box regression model (*Bbox reg.*) under all the tested cases. The improvement on both of them is on average 7 points. Our models also improve for the mAP with IoU≥ 0.5 case but with a smaller amount (around 0.7 points). In Figure 8 we plot the mAP as a function of the IoU threshold. We can observe that the improvement on the detection performance thanks to the proposed localization models starts to clearly appear on the 0.65 IoU threshold and then grows wider till the 0.9. In Table 3 we provide the per class AP for the best approach on each AP metric. Also, in Table 4 we report the AP results on VOC2012 test set but only for the IoU threhold of 0.5 since this is the only metric that the evaluation server provides. In this dataset we achieve mAP of 74.8% which is the state-of-the-art at the time of writing this paper (6/11/2015). Finally, in Figure 9 we examine the detection performance behaviour with respect to the number of iterations used by our pipeline. We observe that as we increase the number of iterations, the mAP for high IoU thresholds (e.g. IoU ≥ 0.8) continues to improve while for lower thresholds the improvements stop on the first two iterations.

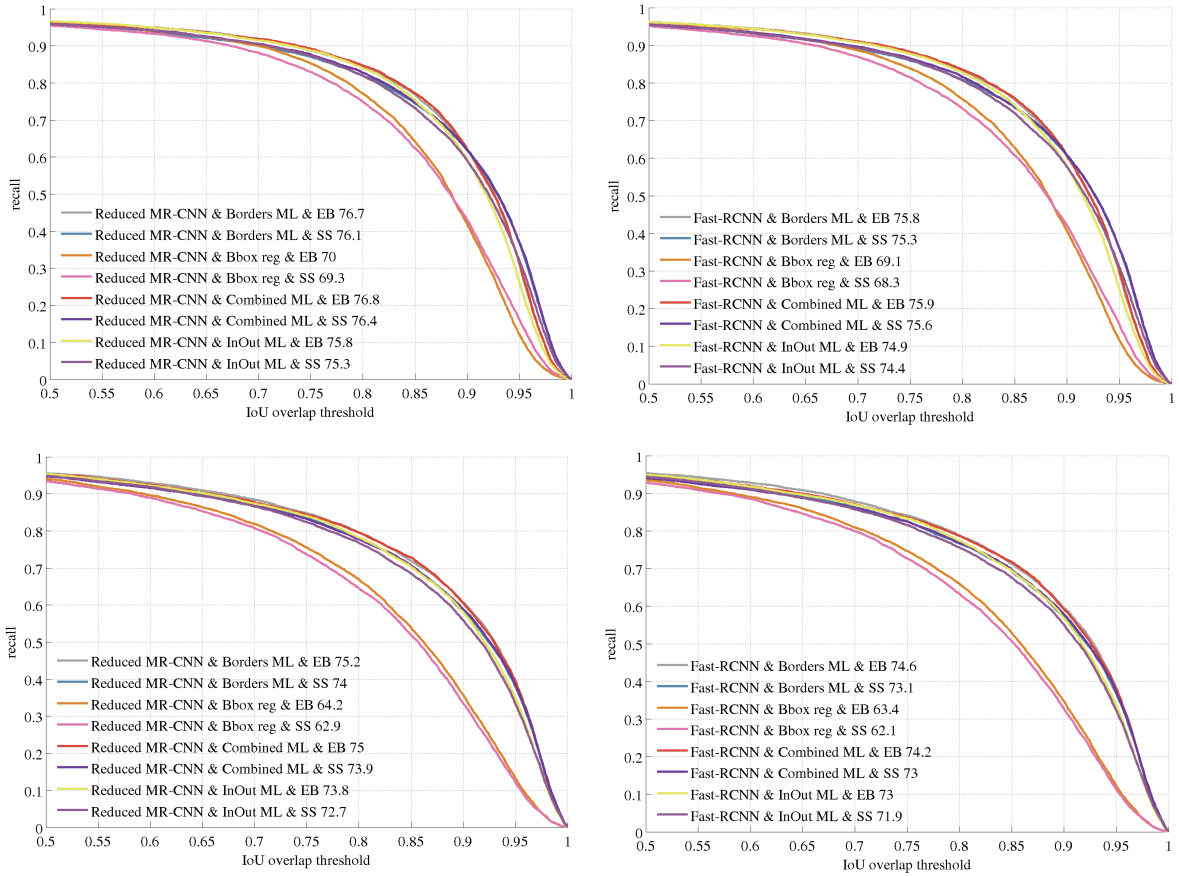


Figure 7: Recall of ground truth bounding boxes as a function of the IoU thresholds. Note that, because we perform class-specific localization the recall that those plots report is obtained after averaging the per class recalls. ***Top-Left:*** Recalls for the *Reduced MR-CNN* model after one iteration of the detection pipeline. ***Bottom-Left:*** Recalls for the *Reduced MR-CNN* model after four iterations of the detection pipeline.***Top-Right:*** Recalls for the *Fast-RCNN* model after one iteration of the detection pipeline. ***Bottom-Right:*** Recalls for the *Fast-RCNN* model after four iterations of the detection pipeline.

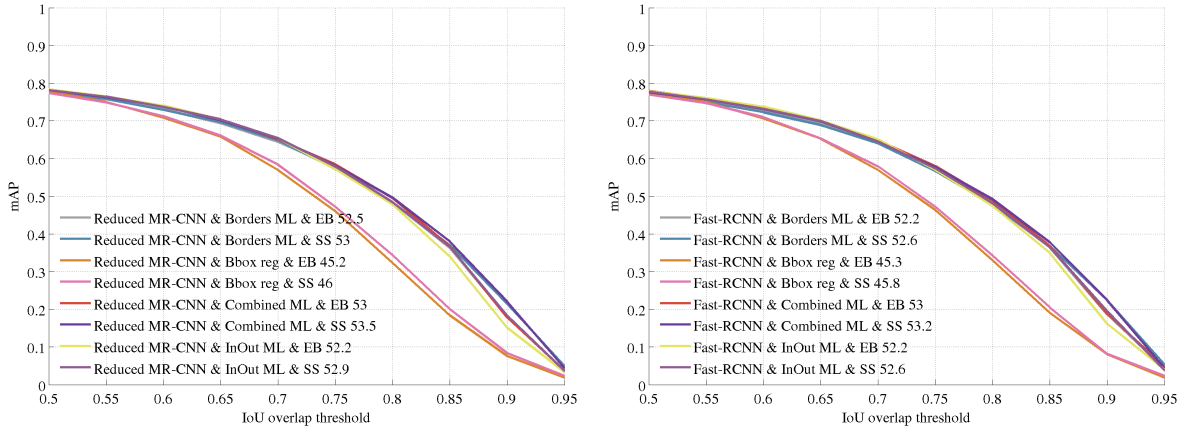


Figure 8: mAP as a function of the IoU threshold. The mAP is evaluated on PASCAL VOC2007. ***Left plot:*** includes the configurations with the *Reduced-MR-CNN* recognition model. ***Right plot:*** includes the configurations with the *Fast-RCNN* recognition model.

Detection Pipeline			mAP		
Localization	Recognition	Initial Boxes	IoU ≥ 0.5	IoU ≥ 0.7	*COCO* style
–	*Reduced-MR-CNN*	2*k Edge Box*	0.747	0.434	0.362
InOut ML	*Reduced-MR-CNN*	2*k Edge Box*	0.783	**0.654**	0.522
Borders ML	*Reduced-MR-CNN*	2*k Edge Box*	0.780	0.644	0.525
Combined ML	*Reduced-MR-CNN*	2*k Edge Box*	**0.784**	0.650	0.530
Bbox reg.	*Reduced-MR-CNN*	2*k Edge Box*	0.777	0.570	0.452
–	*Reduced-MR-CNN*	2*k Sel. Search*	0.719	0.456	0.368
InOut ML	*Reduced-MR-CNN*	2*k Sel. Search*	0.782	**0.654**	0.529
Borders ML	*Reduced-MR-CNN*	2*k Sel. Search*	0.777	0.648	0.530
Combined ML	*Reduced-MR-CNN*	2*k Sel. Search*	0.781	0.653	**0.535**
Bbox reg.	*Reduced-MR-CNN*	2*k Sel. Search*	0.774	0.584	0.460
–	*Fast-RCNN*	2*k Edge Box*	0.729	0.427	0.356
InOut ML	*Fast-RCNN*	2*k Edge Box*	0.779	0.651	0.522
Borders ML	*Fast-RCNN*	2*k Edge Box*	0.774	0.641	0.522
Combined ML	*Fast-RCNN*	2*k Edge Box*	0.780	0.648	0.530
Bbox reg.	*Fast-RCNN*	2*k Edge Box*	0.773	0.570	0.453
–	*Fast-RCNN*	2*k Sel. Search*	0.710	0.446	0.362
InOut ML	*Fast-RCNN*	2*k Sel. Search*	0.777	0.645	0.526
Borders ML	*Fast-RCNN*	2*k Sel. Search*	0.772	0.640	0.526
Combined ML	*Fast-RCNN*	2*k Sel. Search*	0.775	0.645	0.532
Bbox reg.	*Fast-RCNN*	2*k Sel. Search*	0.769	0.579	0.458

Table 2: mAP results on VOC2007 test set for IoU thresholds of 0.5 and 0.7 as well as the COCO style of measuring the AP which actually averages traditional AP of PASCAL for various thresholds between 0.5 and 1 (specifically the thresholds 0.5:0.05:95 are being used). The reported results are obtained after running the detection pipeline for $T = 4$ iterations. With hyphen (–) we signify not using at all the localization models which means that the pipeline runs only for $t = 1$ iteration.

Metric	Approach	areo	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mean
IoU ≥ 0.5	*Reduced-MR-CNN & Combined ML & EB*	0.804	0.855	0.776	0.729	0.622	0.868	0.875	0.886	0.613	0.860	0.739	0.861	0.870	0.826	0.791	0.517	0.794	0.752	0.866	0.777	0.784
IoU ≥ 0.7	*Reduced-MR-CNN & In Out ML & EB*	0.707	0.742	0.622	0.481	0.452	0.840	0.747	0.786	0.429	0.730	0.670	0.754	0.779	0.669	0.581	0.309	0.655	0.693	0.736	0.690	0.654
COCO style	*Reduced-MR-CNN & Combined ML & SS*	0.580	0.603	0.500	0.413	0.367	0.703	0.631	0.661	0.357	0.581	0.500	0.620	0.625	0.545	0.494	0.269	0.522	0.579	0.602	0.555	0.535

Table 3: AP results of each class on VOC2007. Only the best approach, as reported in Table 2 is included here for brevity.

Approach	areo	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mean
Ours: *Reduced-MR-CNN & In Out ML & EB*	0.863	0.830	0.761	**0.608**	0.546	**0.799**	0.790	**0.906**	0.543	**0.816**	0.620	**0.890**	**0.857**	**0.855**	**0.828**	0.497	**0.766**	0.675	**0.832**	0.674	**0.748**
Ours: *Reduced-MR-CNN & Borders ML & EB*	0.865	0.827	0.755	0.602	0.535	0.791	0.785	0.902	0.533	0.800	0.607	0.886	**0.857**	0.848	0.826	0.496	0.765	0.673	0.831	0.676	0.743
Ours: *Reduced-MR-CNN & Combined ML & EB*	**0.866**	**0.834**	0.765	0.604	0.544	0.798	0.786	0.902	0.546	0.810	0.618	0.889	**0.857**	0.847	**0.828**	**0.498**	0.763	0.678	0.830	0.679	0.747
MR-CNN [7]	0.855	0.829	**0.766**	0.578	**0.627**	0.794	0.772	0.866	**0.550**	0.791	**0.622**	0.870	0.834	0.847	0.789	0.453	0.734	0.658	0.803	**0.740**	0.739
HyperNet_VGG	0.842	0.785	0.736	0.556	0.537	0.787	**0.798**	0.877	0.496	0.749	0.521	0.860	0.817	0.833	0.818	0.486	0.735	0.594	0.799	0.657	0.714
HyperNet_SP	0.841	0.783	0.734	0.555	0.536	0.786	0.796	0.875	0.495	0.749	0.521	0.856	0.816	0.832	0.816	0.484	0.732	0.593	0.797	0.656	0.713
Faster R-CNN [26]	0.849	0.798	0.743	0.539	0.498	0.775	0.759	0.885	0.456	0.771	0.553	0.869	0.817	0.809	0.796	0.401	0.726	0.609	0.812	0.615	0.704
Fast R-CNN & YOLO [25]	0.830	0.785	0.737	0.558	0.431	0.783	0.730	0.892	0.491	0.743	0.566	0.872	0.805	0.805	0.747	0.421	0.708	0.683	0.815	0.670	0.704
Deep Ensemble COCO [10]	0.840	0.794	0.716	0.519	0.511	0.741	0.721	0.886	0.483	0.734	0.578	0.861	0.800	0.807	0.704	0.466	0.696	**0.688**	0.759	0.714	0.701
NoC [27]	0.828	0.790	0.716	0.523	0.537	0.741	0.690	0.849	0.469	0.743	0.531	0.850	0.813	0.795	0.722	0.389	0.724	0.595	0.767	0.681	0.688
Fast R-CNN [8]	0.823	0.784	0.708	0.523	0.387	0.778	0.716	0.893	0.442	0.730	0.550	0.875	0.805	0.808	0.720	0.351	0.683	0.657	0.804	0.642	0.684
UMICH_FGS_STRUCT [34]	0.829	0.761	0.641	0.446	0.494	0.703	0.712	0.846	0.427	0.686	0.558	0.827	0.771	0.799	0.687	0.414	0.690	0.600	0.720	0.662	0.664

Table 4: The object detection leaderboard of PASCAL VOC2012 test set. The AP results that are reported here are with IoU ≥ 0.5. For brevity we report only the top-10 methods.

Detection Pipeline			mAP		
Localization	Recognition	Initial Boxes	IoU ≥ 0.5	IoU ≥ 0.7	*COCO* style
–	*Reduced-MR-CNN*	10*k Sliding Windows*	0.617	0.174	0.227
InOut ML	*Reduced-MR-CNN*	10*k Sliding Windows*	0.770	0.633	0.513
Borders ML	*Reduced-MR-CNN*	10*k Sliding Windows*	0.764	0.626	0.513
Combined ML	*Reduced-MR-CNN*	10*k Sliding Windows*	0.773	0.639	0.521
Bbox reg.	*Reduced-MR-CNN*	10*k Sliding Windows*	0.761	0.550	0.436

Table 5: In this table we show mAP results for VOC2007 test set when using *sliding windows* as initial set of candidate boxes. In order to generate the sliding windows we use the publicly available code that accompanies the work of Hosang *et al*. [13] that includes a sliding window implementation inspired by *BING* [3, 35]).

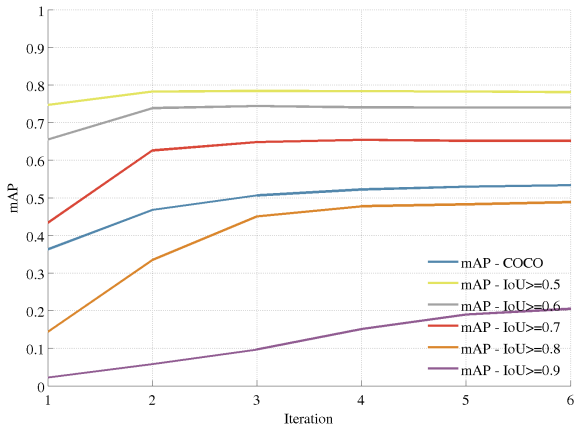


Figure 9: Plot of the mAP as a function of the iterations number of our detection pipeline. The model used for this plot is the *Reduced-MR-CNN* recognition model with the *In-Out ML* localization model and Edge Box proposals.

5.3. Sliding windows as initial set of candidate boxes

In Table 5 we provide the detection accuracy of our pipeline when, for generating the initial set of candidate boxes, we use a simple sliding windows scheme (of $10k$ windows per image). We observe that:

- Even in this case, our pipeline achieves very high mAP results that are close to the ones obtained with selective search or edge box proposals. We emphasize that this is true even for the IoU≥ 0.7 or the COCO style of mAP that favour better localized detections, despite the fact that in the case of sliding windows the initial set of candidate boxes is considerably less accurately localized than in the edge box or in the selective search cases (see Table 1).

- In the case of sliding windows, just scoring the candidate boxes with the recognition model (hyphen (–) case) yields much worse mAP results than the selective search or the edge box proposals case. However, when we use the full detection pipeline that includes localization models and re-scoring of the new better localized candidate boxes, then this gap is significantly reduced.

- The difference in the mAP results between the proposed localization models (*In-Out ML*, *Borders ML*, and *Combined ML*) and the *bounding box regression* model (*Bbox reg.*) is even greater in the case of sliding windows.

To the best of our knowledge, the above mAP results are considerably higher than those of any other detection method when only sliding windows are used for the initial bounding box proposals (similar experiments are reported in [8, 13]). We also note that we had not experimented with increasing the number of sliding windows. Furthermore, the tested recognition model and localization models were not re-trained with sliding windows in the training set. As a result, we foresee that by exploring those two factors one might be able to further boost the detection performance for the sliding windows case.

5.4. Qualitative results

In Figure 10 we provide sample qualitative results that compare the newly proposed localization models (*In-Out ML*, *Borders ML*, and *Combined ML*) with the current state-of-the-art *bounding box regression* localization model.

6. Conclusion

We proposed a novel object localization methodology that is based on assigning probabilities related to the localization task on each row and column of the region in which it searches the object. Those probabilities provide useful information regarding the location of the object inside the search region and they can be exploited in order to infer its boundaries with high accuracy.

We implemented our model via using a convolutional neural network architecture properly adapted for this task, called LocNet, and we extensively evaluated it on PASCAL VOC2007 test set. We demonstrate that it outperforms CNN-based bounding box regression on all the evaluation metrics and it leads to a significant improvement on those metrics that reward good localization. Importantly, LocNet can be easily plugged into existing state-of-the-art object detection methods, in which case we show that it contributes to significantly boosting their performance. Finally, we demonstrate that our object detection methodology can achieve very high mAP results even when the initial set of candidate boxes is generated by a simple sliding windows scheme.

References

[1] B. Alexe, T. Deselaers, and V. Ferrari. Measuring the objectness of image windows. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(11), 2012. 2

[2] P. Arbeláez, J. Pont-Tuset, J. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. In *Computer Vision and Pattern Recognition*, 2014. 2

[3] M.-M. Cheng, Z. Zhang, W.-Y. Lin, and P. Torr. Bing: Binarized normed gradients for objectness estimation at 300fps. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*. IEEE, 2014. 9

[4] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge 2007 (voc 2007) results (2007), 2008. 2

[5] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge 2012, 2012. 2

[6] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 2010. 2, 3, 4

[7] S. Gidaris and N. Komodakis. Object detection via a multi-region & semantic segmentation-aware cnn model. *arXiv preprint arXiv:1505.01749*, 2015. 2, 3, 4, 7, 9, 12

[8] R. Girshick. Fast r-cnn. *arXiv preprint arXiv:1504.08083*, 2015. 2, 4, 7, 9, 10, 12

[9] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, 2014. 2, 7, 11, 12

[10] J. Guo and S. Gould. Deep CNN ensemble with data augmentation for object detection. *CoRR*, abs/1506.07224, 2015. 9

[11] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *arXiv preprint arXiv:1406.4729*, 2014. 2, 5

[12] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *arXiv preprint arXiv:1502.01852*, 2015. 3

[13] J. Hosang, R. Benenson, P. Dollár, and B. Schiele. What makes for effective detection proposals? *arXiv preprint arXiv:1502.05082*, 2015. 4, 7, 9, 10

[14] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 3

[15] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014. 6

[16] P. Krähenbühl and V. Koltun. Learning to propose objects. 2

[17] P. Krähenbühl and V. Koltun. Geodesic object proposals. In *Computer Vision–ECCV 2014*. Springer, 2014. 2

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 2012. 3

[19] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1989. 3

[20] K. Lenc and A. Vedaldi. R-cnn minus r. *arXiv preprint arXiv:1506.06981*, 2015. 3

[21] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014*. 2014. 1

[22] W. Ouyang, P. Luo, X. Zeng, S. Qiu, Y. Tian, H. Li, S. Yang, Z. Wang, Y. Xiong, C. Qian, et al. Deepid-net: multi-stage and deformable deep convolutional neural networks for object detection. *arXiv preprint arXiv:1409.3505*, 2014. 2

[23] G. Papandreou, I. Kokkinos, and P.-A. Savalle. Modeling local and global deformations in deep learning: Epitomic convolution, multiple instance learning, and sliding window detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015. 3

[24] P. O. Pinheiro, R. Collobert, and P. Dollar. Learning to segment object candidates. *arXiv preprint arXiv:1506.06204*, 2015. 2, 4

[25] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. *arXiv preprint arXiv:1506.02640*, 2015. 3, 9

[26] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*, 2015. 2, 3, 9, 12

[27] S. Ren, K. He, R. Girshick, X. Zhang, and J. Sun. Object detection networks on convolutional feature maps. *arXiv preprint arXiv:1504.06066*, 2015. 2, 9

[28] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013. 2, 3

[29] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 3, 5, 6, 12

[30] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*, 2014. 3

[31] C. Szegedy, S. Reed, D. Erhan, and D. Anguelov. Scalable, high-quality object detection. *arXiv preprint arXiv:1412.1441*, 2014. 2

[32] C. Szegedy, A. Toshev, and D. Erhan. Deep neural networks for object detection. In *Advances in Neural Information Processing Systems*, 2013. 2

[33] K. E. Van de Sande, J. R. Uijlings, T. Gevers, and A. W. Smeulders. Segmentation as selective search for object recognition. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, 2011. 2, 3, 6

[34] Z. Yuting, S. Kihyuk, V. Ruben, P. Gang, and H. Lee. Improving object detection with deep convolutional networks via bayesian optimization and structured prediction. *arXiv preprint arXiv:1504.03293*, 2015. 2, 9

[35] Q. Zhao, Z. Liu, and B. Yin. Cracking bing and beyond. In *Proceedings of the British Machine Vision Conference. BMVA Press*, 2014. 9

[36] Y. Zhu, R. Urtasun, R. Salakhutdinov, and S. Fidler. segdeepm: Exploiting segmentation and context in deep neural networks for object detection. *arXiv preprint arXiv:1502.04275*, 2015. 2

[37] C. L. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. In *Computer Vision–ECCV 2014*. 2014. 2, 3, 6, 7

A. Bounding box regression model

This localization model consists of four CNN-based scalar regression functions $f_x(B, c)$, $f_y(B, c)$, $f_w(B, c)$, and $f_h(B, c)$ that given a candidate box B and a category c, they actually predict the coefficients of a geometric transformation that will ideally map the candidate box B to a ground truth bounding box of the c object category [9]. Specifically, if $B = (B_x, B_y, B_w, B_h)$ are the coordinates

of the candidate box in form of its top-left corner (B_x, B_y) and its width and height (B_w, B_h), then the predicted candidate box $\hat{B}$ is given by the following equations:

$$\hat{B}_x = B_w \cdot f_x(B, c) + B_x \quad (9)$$

$$\hat{B}_y = B_h \cdot f_y(B, c) + B_y \quad (10)$$

$$\hat{B}_w = B_w \cdot exp(f_w(B, c)) \quad (11)$$

$$\hat{B}_h = B_h \cdot exp(f_h(B, c)). \quad (12)$$

Hence, the four scalar target regression values $T = \{t_x, t_y, t_w, t_h\}$ for the ground truth bounding box $B^{gt} = (B^{gt}_x, B^{gt}_y, B^{gt}_w, B^{gt}_h)$ are defined as:

$$t_x = \frac{B^{gt}_x - B_x}{B_w} \qquad t_y = \frac{B^{gt}_y - B_y}{B_h} \quad (13)$$

$$t_w = log(\frac{B^{gt}_w}{B_w}) \qquad t_h = log(\frac{B^{gt}_h}{B_h})). \quad (14)$$

For the CNN architecture that implements the bounding box regression model we adopt the one proposed in [7]. As a loss function we use the sum of euclidean distances between the target values and the predicted values of each training sample. The final fully connected layer is initialized from a Gaussian distribution with standard deviation of 0.01. The rest training details (*i.e*. SGD, mini-batch, definition of training samples) are similar to those described for the proposed localization models.

B. Training the localization models

As proposed in Fast-RCNN [8], when training the *bounding box regression* model we simultaneously train the Fast-RCNN recognition model with the two models sharing their convolutional layers In our experiments, this way of training improves the accuracy of both the Fast-RCNN recognition model and the bounding box regression model.

On the contrary, the newly proposed localization models (*i.e*., *Borders ML*, *In-Out ML*, and *Combined ML*) are not currently trained simultaneously with the recognition model. We expect that the joint training of these models with the recognition model can help to further improve their overall performance, which is therefore something that we plan to explore in the future.

C. Recognition models

Reduced MR-CNN model: We based the implementation of this model on the state-of-the-art MR-CNN detection system that was proposed in [7]. Briefly, the MR-CNN detection system recognises a candidate box by focusing on multiple regions of it, each with a dedicated network component termed as region adaptation module. In our implementation however, for efficiency reasons and in order to speed up the experiments, we applied the following reductions:

- We include only six out of the ten regions proposed, by skipping the half regions.
- We do not include the semantic segmentation-aware CNN features.
- We reduce the total amount parameters on the region adaptation modules.

In order to achieve the reduction of parameters on the hidden fully connected layers fc6 and fc7 of the region adaptation modules, each of them is decomposed on two fully connected layers without any non-linearities between them. Specifically, the fc6 layer with weight matrix $W_6 : 25088 \times 4096$ is decomposed on the layers fc6_1 and fc6_2 with weight matrices $W_{6.1} : 25088 \times 1024$ and $W_{6.2} : 1024 \times 4096$ correspondingly. The fc7 layer with weight matrix $W_7 : 4096 \times 4096$ is decomposed on the layers fc7_1 and fc7_2 with weight matrices $W_{7.1} : 4096 \times 256$ and $W_{7.2} : 256 \times 4096$ correspondingly. To train the Reduced MR-CNN network, we first train only the original candidate box region of it without reducing the parameters of the fc6 and fc7 layers. Then, we apply the truncated SVD decomposition on the aforementioned layers (for more details see section §3.1 of [8]) that results on the layers fc6_1, fc6_2, fc7_1, and fc7_2. We copy the parameters of the resulting fully connected layers to the corresponding layers of the rest region adaptation modules of the model and we continue training.

Fast-RCNN model: We re-implemented Fast-RCNN based on the publicly available code provided from Fast-RCNN [8] and Faster-RCNN [26]. Here we will describe only the differences of our implementation with the original Fast-RCNN system [8]. In our implementation, we have different branches for the recognition of a candidate box and for its bounding box regression after the last convlutional layer (conv5_3 for the VGG16-Net [29]) that do not share any weights. In contrary, the original Fast-RCNN model splits to two branches after the last hidden layer. We applied this modification because, in our case, the candidate box that is fed to the regression branch is enlarged by a factor $\alpha = 1.3$ while the candidate box that is fed to recognition branch is not. Also, after the fine-tuning, we remove the softmax layer of the recognition branch and we train linear SVMs on top of the features that the last hidden layer of the recognition branch yields, just as R-CNN [9] does. Finally, we do not reduce the parameters of the fully connected layers by applying the truncated SVD decomposition on them as in the original paper. In our experiments those changes improved the detection performance of the model.

D. Object detection pipeline

In Algorithm 1 of section §2 we provide the pseudo-code of the object detection pipeline that we adopt. For clar-

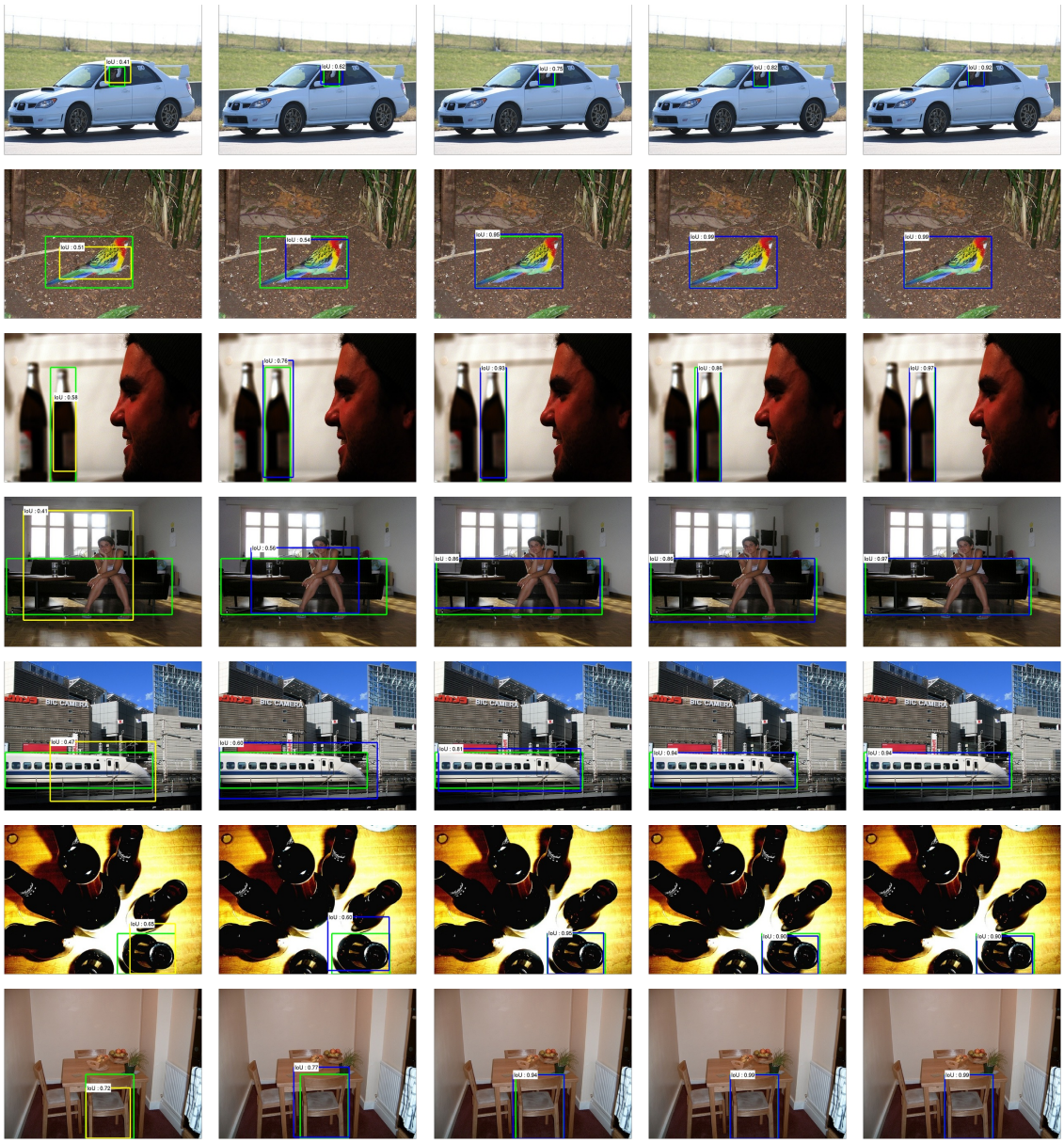
Algorithm 2: Object detection pipeline

Input : Image $\mathbf{I}$, initial set of candidate boxes $\{\mathbf{B}_c^1\}_{c=1}^C$

Output: Final list of per category detections $\{\mathbf{Y}_c\}_{c=1}^C$

for $t \leftarrow 1$ ***to*** T **do**
  for $c \leftarrow 1$ ***to*** C **do**
    $\mathbf{S}_c^t \leftarrow$ *Recognition*$(\mathbf{B}_c^t|\mathbf{I}, c)$
    if $t == 1$ **then**
      $\{\mathbf{S}_c^t, \mathbf{B}_c^t\} \leftarrow PruneCandidateBoxes(\{\mathbf{S}_c^t, \mathbf{B}_c^t\})$
    end
  end
  if $t < T$ **then**
    for $c \leftarrow 1$ ***to*** C **do**
      $\mathbf{B}_c^{t+1} \leftarrow$ *Localization*$(\mathbf{B}_c^t|\mathbf{I}, c)$
    end
  end
end
for $c \leftarrow 1$ ***to*** C **do**
  $\mathbf{D}_c \leftarrow \cup_{t=1}^T \{\mathbf{S}_c^t, \mathbf{B}_c^t\}$
  $\mathbf{Y}_c \leftarrow$ *PostProcess*$(\mathbf{D}_c)$
end

ity purposes, the pseudo-code that is given corresponds to the single object category detection case and not the multiple object categories case that we are dealing with. Moreover, the actual detection algorithm, after scoring the candidate boxes for the first time $t = 1$, it prunes the candidate boxes with low confidence in order to reduce the computational burden of the subsequent iterations. For that purpose, we threshold the candidate boxes of each category such that their average number per image and per category to be around 18 boxes. Also, during this step, non-max-suppression with IoU of 0.95 is applied in order to remove near duplicate candidate boxes (in the case of using sliding windows to generate the initial set of candidate boxes this IoU threshold is set to 0.85). A more detailed algorithm of our detection pipeline is presented in Algorithm 2. Note that, since the initial candidate boxes $\{\mathbf{B}_c^1\}_{c=1}^C$ are coming from a category-agnostic bounding box proposal algorithm, those boxes are the same for all the categories and when applying on them (during $t = 1$ iteration) the recognition module, the computation between all the categories can be shared.



(a) Candidate box **(b)** Bbox regression **(c)** In-Out ML **(d)** Borders ML **(e)** Combined ML

Figure 10: Qualitative results of the bounding box localization step given an initial candidate box (column *(a)*) from the bounding box regression model (column *(b)*), the *In-Out ML* localization model (column *(c)*), the *Borders ML* localization model (column *(d)*), and the *Combined ML* localization model (column *(e)*). The candidate box is drawn with yellow color, the predicted boxes are drawn with blue color, and the ground truth bounding box is drawn with green color.